Informe de Calidad (Sprint 1)

Autores: Aarón Rodríguez y Manuel Caballero

ANÁLISIS 28 OCTUBRE 2020

CAPTURA

INCIDENCIAS

El análisis pasa los criterios de calidad de la organización ya que como se puede observar todos los criterios tienen una calificación A. Sin embargo, a pesar de que la calificación de *code smells* sea A, se pueden detectar un importante número de ellos a corregir.

Existen 24 *code smells* que suman casi 3 horas de deuda técnica y se encuentran principalmente en la clase MainActivity y DetailActivity, con 13 y 7 respectivamente. Si nos fijamos en tiempo de deuda técnica, observamos que realmente se concentra en la clase MainActivity con algo más de 2 horas. Por otro lado, vemos la existencia de 2 code smells con severidad *critical* y otros dos con severidad *major*.

PLAN DE ACCIÓN

1) Arreglar los *code smells critical* que entre ambos suman 20 minutos. Se encuentran en la clase MainActivity y ParserJSONGasolineras.
2) Solucionar los *code smells major* que suman 10 minutos. Se encuentran en las clases ParserJSONGasolineras y RemoteFetch.
3) Arreglar dos *code smells minor* que se encuentran en la clase MainActivity y entre los dos suman 15 minutos. Uno indica “Use the primitive boolean expression here” y el otro “Extract this nested code block into a method.”

Comentarios:

Si bien el análisis pasa todos los criterios de calidad de la organización, las acciones seleccionadas tienen un objetivo de prevención y de mejorar la calidad del código.

Informe de Calidad (Sprint 1)

Autores: Aarón Rodríguez y Manuel Caballero

ANÁLISIS 01 NOVIEMBRE 2020

CAPTURA

INCIDENCIAS

El análisis pasa los criterios de calidad de la organización ya que como se puede observar todos los criterios tienen una calificación A. Sin embargo, a pesar de que la calificación de *code smells* sea A, se pueden detectar un importante número de ellos a corregir.

Existen 22 *code smells* que suman casi 3 horas de deuda técnica y se encuentran principalmente en la clase MainActivity y DetailActivity, con 13 y 7 respectivamente. Si nos fijamos en tiempo de deuda técnica, observamos que realmente se concentra en la clase MainActivity con algo más de 2 horas. También existe 1 *code smell* en la clase de test ExampleInstrumentedTest y otro en la clase ParserJSONGasolineras. Todos los *code smells* detectados son de severidad baja: *minor* e *info*.

PLAN DE ACCIÓN

1) Solucionar los *code smells* con severidad *minor* presentes en la clase MainActivity. En orden por tiempo de duda técnica, uno indica “Make presenterGasolineras a static final constant or non-public and provide accessors if needed.” con un tiempo de 10 minutos y otro de 5 minutos que indica “Move this method into "CargaDatosGasolinerasTask"”. Los otros dos errores muestran la necesidad de eliminar *imports* innecesarios.
2) Arreglar 6 *code smells info* que se encuentran en la clase DetailActivity y que corresponde a un mismo error repetido en la declaración de varios botones: “Declare "txtDireccion" on a separate line”. Por otro lado, arreglar el *code smell minor* que indica la existencia de un import innecesario.
3) Solucionar un *code smell* en la clase DetailActivity que indica la necesidad de eliminar import innecesario y otro *issue* en la clase ParserJSONGasolineras “Replace charset name argument with StandardCharsets.UTF_8”. Así, ambas clases quedarán sin ningún *code smell*.

Comentarios:

Si bien el análisis pasa todos los criterios de calidad de la organización, las acciones seleccionadas tienen un objetivo de prevención y de mejorar la calidad del código.

Informe de Calidad (Sprint 1)

Autores: Aarón Rodríguez y Manuel Caballero

ANÁLISIS 02 NOVIEMBRE 2020

CAPTURA

INCIDENCIAS

El análisis pasa los criterios de calidad de la organización ya que como se puede observar todos los criterios tienen una calificación A. Sin embargo, a pesar de que la calificación de *code smells* sea A, se pueden detectar un importante número de ellos a corregir.

Existen 22 *code smells* que suman algo más de 3 horas de deuda técnica y se encuentran principalmente en la clase MainActivity, seguido de Gasolinera y NoDatosActivity, con 12,3 y 3 respectivamente. Si nos fijamos en tiempo de deuda técnica, observamos que realmente se concentra en la clase MainActivity con algo más de 2 horas. También existe 1 *code smell* en la clase de test PresenterGasolineras, otro en DetailActivity y en la clase ExampleUnitTest. El *issue* presente en esta última clase es de severidad major, sin embargo, todos los demás *code smells* detectados son de severidad baja: *minor* e *info*.

PLAN DE ACCIÓN

1) Solucionar el *code smell* con severidad *major* presente en la clase ExampleUnitTest, que indica la conveniencia de usar assertEquals() en lugar de assertTrue().
2) Arreglar 2 *code smells minor* que se encuentran en la clase MainActivity. Uno indica “Make listViewGasolineras a static final constant or non-public and provide accessors if needed” y el otro “Make adapter a static final constant or non-public and provide accessors if needed”. Entre los dos suman 20 minutos de deuda técnica y corresponden a un mismo error repetido.
3) Solucionar dos *code smells info* en la clase MainActivity que indican un mismo error: “Don't override a deprecated method or explicitly mark it as "@Deprecated"”. Finalmente, solucionar 3 *code smells minor* presentes en la clase NoDatosActivity que corresponde a imports no usados.

Comentarios:

Si bien el análisis pasa todos los criterios de calidad de la organización, las acciones seleccionadas tienen un objetivo de prevención y de mejorar la calidad del código.

Informe de Calidad (Sprint 1)

Autores: Aarón Rodríguez y Manuel Caballero

ANÁLISIS 03 NOVIEMBRE 2020

CAPTURA

INCIDENCIAS

El análisis pasa los criterios de calidad de la organización ya que como se puede observar todos los criterios tienen una calificación A. Sin embargo, a pesar de que la calificación de *code smells* sea A, se pueden detectar un importante número de ellos a corregir.

Existen 18 *code smells* que suman algo menos de 3 horas de deuda técnica y se encuentran principalmente en la clase MainActivity con 12, seguido de Gasolinera con 3 y DetailActivity, PresenterGasolinera y ExampleUnitTest con 1 *issue* cada uno. Si nos fijamos en tiempo de deuda técnica, observamos que realmente se concentra en la clase MainActivity con algo más de 2 horas. Podemos observar que en la mayor parte de code smells son de severidad info y añaden una gran cantidad de deuda técnica, pero muchos de ellos hacen referencia a uso de métodos o clases deprecated, que no solucionaremos en el plan de acción debido la falta una alternativa válida a estos.

PLAN DE ACCIÓN

1) Solucionar los *code smell* con severidad *info* presentes en la clase Gasolinera, que corresponde a líneas de código comentadas que deberían ser eliminadas.
2) Arreglar 3 *code smells info* que se encuentran en la clase MainActivity. Uno indica “Remove this unnecessary cast to "ArrayList".” y los otros dos indicant la conveniencia de eliminar código comentado. Entre los tres suman 15 minutos de deuda técnica.
3) Solucionar un *code smells minor* en la clase PresenterGasolineras que indica la presencia de un import no usado y solucionar también un *code smell minor* en la clase DetailActivity que indica que se debería eliminar código que se encuentra comentado.

Comentarios:

Si bien el análisis pasa todos los criterios de calidad de la organización, las acciones seleccionadas tienen un objetivo de prevención y de mejorar la calidad del código.

Informe de Calidad (Sprint 1)

Autores: Aarón Rodríguez y Manuel Caballero

ANÁLISIS 06 NOVIEMBRE 2020 (MAÑANA)

CAPTURA

INCIDENCIAS

El análisis no pasa los criterios de calidad de la organización ya que como se puede observar los code smells han aumentado considerablemente al juntar todo el código relativo al sprint 1. Ahora al haber 68 code smells tenemos una deuda técnica de 6 horas y 31 minuto. Sin embargo, a pesar de todo esto la calificación de code smells es A.

Los code smells se encuentran principalmente en la clase mainActivity y Gasolinera. Si nos fijamos en tiempo de deuda técnica, observamos que se concentra en la clase MainActivity con algo más de 2 horas y media, resaltarDescuento con 1 hora. En cuanto a la severidad de los code smells están bastante repartidos, ya que hay 8 críticos, 17 mayores, 26 menores y 17 de info.

PLAN DE ACCIÓN

1) Solucionar los code smells de severidad crítica en los que hay que cambiar el método replaceAll() por replace().
2) Solucionar los code smells de severidad mayor en los que hay que hacer un swap en los argumentos de los métodos de los tests.
3) Solucionar los code smells de severidad minor en los que hay que eliminar imports tanto duplicados como sin usar.
4) Solucionar los code smells de severidad minros en los que hay que renombrar variables.
5) Solucionar los *code smells minor* en la clase PresenterGasolineras que indica la presencia de un import no usado y solucionar también los *code smell minor* en la clase DetailActivity que indica que se debería eliminar código que se encuentra comentado.

Comentarios:

Con las acciones planteadas en este informe se esperaba reducir considerablemente la deuda técnica de la aplicación con el fin de pasar los tests de calidad de sonar.

Informe de Calidad (Sprint 1)

Autores: Aarón Rodríguez y Manuel Caballero

ANÁLISIS 06 NOVIEMBRE 2020 (TARDE)

CAPTURA

INCIDENCIAS

El análisis pasa los criterios de calidad de la organización ya que como se puede observar todos los criterios tienen una calificación A. Sin embargo, a pesar de que la calificación de *code smells* sea A, se pueden detectar un importante número de ellos a corregir.

Existen 26 *code smells* que suman algo menos de 4 horas de deuda técnica y se encuentran principalmente en la clase MainActivity con 14, seguido de Gasolinera con 5 y CalculaGasolineraMasBarataITest, ResaltarDescuentosTest y Distancia con 2 *issue* cada uno. Finalmente, la clase PruebasUnitariasTest tiene un solo *code smell*. Si nos fijamos en tiempo de deuda técnica, observamos que realmente se concentra en la clase MainActivity con algo más de 2 horas. Hay un total de 4 *code smells* con severidad *critical* y 2 con severidad *major*. Estos *issues* con una mayor severidad están concentrados en dos clases: MainActivity y Distancia.

PLAN DE ACCIÓN

1) Solucionar los *code smell* con severidad *critical* presentes en la clase MainActivity. Uno de ellos corresponde a la necesidad de refactorizar un método con una complejidad demasiado elevada, otro indica “Use static access with "androidx.core.content.ContextCompat" for "checkSelfPermission"” y el último muestra el mismo error en otro caso concreto “Use static access with "com.google.android.gms.common.api.CommonStatusCodes" for "RESOLUTION_REQUIRED"”. De esta forma eliminaríamos los *issues* de mayor severidad presentes en la clase.
2) Arreglar 2 *code smells* que se encuentran en la clase Distancia. Uno tiene severidad *critical* e indica “Rename this constant name to match the regular expression '^[A-Z][A-Z0-9]*(_[A-Z0-9]+)*$'.” y el otro es s¡de severiad *major* indica la conveniencia de crear un constructor privado para esconder el público. Así, la clase Distancia no tendría ningún *code smell*.
3) Solucionar un *code smells major* en la clase PruebasUnitariasTest que indica la conveniencia de hacer uso de AssertEquals para comparar dos valores en lugar de AssertTrue. Después de solucionar este *issue,* la clase PruebasUnitariasTest no tendrá ningún *code smel*l y el proyecto estará libre de *issues* con severidad *major* y *critical*.

Comentarios:

Si bien el análisis pasa todos los criterios de calidad de la organización, las acciones seleccionadas tienen un objetivo de prevención y de mejorar la calidad del código.